多目的非線形最適化手法の一提案 : Vector Simplex法 (最適化の数理科学)

AUTHOR(S):

阪井, 節子; 高浜, 徹行

CITATION:

阪井, 節子 ...[et al]. 多目的非線形最適化手法の一提案 : Vector Simplex法 (最適化の数理科学). 数理解析研究所講究録 2000, 1174: 169-178

ISSUE DATE:

2000-10

URL:

http://hdl.handle.net/2433/64461

数理解析研究所講究録
1174巻 2000年 169-178

多目的非線形最適化手法の一提案
—Vector Simplex 法—

広島修道大学商学部　阪井　節子 (Setsuko Sakai)
Faculty of Commercial Science, Hiroshima Shudo University
広島市立大学情報科学部　高濱　徹行 (Tetsuyuki Takahama)
Faculty of Information Sciences, Hiroshima City University

1　はじめに

現在提案されている多くの線形計画法や非線形計画法は，与えられた条件の下で，総合的な評価尺度を与える単一の目的関数を最小化（あるいは最大化）する最適解を求める方法を提供している．この場合，目的関数の値域は，整数や実数などの全順序関係を持つ全順序集合になる．

一方，多目的計画法は，相競合する評価尺度を持つ複数の目的関数を同時に最小化する解を求める，すなわち，多目的最適化を実現することを目的とする．この場合，目的関数がベクトル値関数となり，関数値の間に半順序関係しか持たない半順序集合を対象とすることになる．したがって，全ての目的関数が最小となる完全最適解が必ずしも存在するとは限らない．このため，多目的計画問題に対して数多くの最適解の解概念が提案され研究されているが，その最も代表的な解の1つがPareto最適解である．

従来，Pareto最適解を求める方法としては，以下のような方法が提案されている[3, 4, 5].

(a) スカラー化手法
もとの多目的最適化問題の各目的関数を総合的に組み合わせて全順序関係を持つ単一目的最適化問題に変換する方法

(b) 目標計画法，妥協計画法
意志決定者が達成したいと考えている目標値からの距離を最小にする解を求める方法

(c) 対話的手法
対話によって得られる局所的な選好情報に基づき意志決定者の選好解を導出する方法

しかし，(a),(b) では，多目的最適化問題から単一目的最適化問題への変換方法によっては，意志決定者の選好が十分に反映されない恐れがあり，(c) では，解の探索の途中の各段階で意志決定者の局所的な選好を取り入れてはいるが，Pareto最適解集合全体を比較するような大局的な選好を取り入れることはできない，などの問題点がある．

多目的最適化問題の解としてPareto最適解を採用した場合の重要な点は，得られたPareto最適解集合の中から，最終的に何らかの評価基準によって実行解を選択する必要があるという点である．したがって，

1. Pareto最適解の全体集合を求める

2. Pareto最適解の全体集合の中から最終的な選好解を選択する

という2段階をとることが必要である．もちろん，(a),(b) でも，パラメータを変化させながら繰り返し問題を解くことによってPareto最適解集合を求めることができるが，目的関数の定義域の次元が増加するにつれ，膨大な計算時間が必要となる．このため，多目的最適化問題を単一目的最適化問題に変換することなく，Pareto最適解集合を直接求める手法が提案されてきている．

既に提案されている手法としては，遺伝的アルゴリズムを利用し，各目的関数について独立に選択を行う方法[6, 7]，解の優越関係に基づいて選択を行う方法[8, 9, 10]，それらを組み合わせた方法[5]などがある．しかし，遺伝的アルゴリズムは通常離散空間を探索するため，実数空間で定義された問題を解く場合

には，必ずしも十分な精度が得られない可能性がある．この問題を解決するためには，実数空間上で直接Pareto最適解集合を求める手法を考案する必要がある．

本研究では，Pareto最適解集合を実数空間上で直接求める手法としてVector Simplex 法を提案する．Vector Simplex 法は，非線形最適化手法の一つである Simplex 法[11, 12, 2]に着目し，Simplex 法の持つ「解候補の集合による探索」という特徴を利用して，Pareto最適解を直接的に求める方法である．Simplex 法は，単一目的最適化問題を解くために，最適解探索の各段階において，解候補集合における最良解，最悪解などを定め，それを基準に鏡映，拡張，収縮などの操作を行い，最悪解を順次改善することにより最適解を求める方法である．多目的最適化問題では，半順序集合を対象とするため，最良解，最悪解などを定めることができるとは限らない．そこで，最良解に対応する「より良い解候補のない解の集合」，最悪解に対応する「より悪い解候補のない解の集合」などを定め，後者の集合の要素を順次改善して行くことによってPareto最適解集合を求める．

以下，2. では多目的最適化問題とPareto最適解を定義し，3. では Simplex 法について説明する．4. では，Vector Simplex 法を提案する．5. では制約のない多目的最適化問題にVector Simplex 法を応用した結果について述べ，6. ではスカラー化法を用いた結果と比較する． 7. はまとめである．

2 多目的最適化問題

一般に，与えられた制約条件の下で，複数個の相競合する目的関数を最小化（あるいは最大化）する問題は多目的最適化問題（あるいは多目的計画問題）と呼ばれる．複数個の目的関数をベクトル値関数として形式化すれば，多目的最適化問題は，ベクトル値最小化問題として以下のように定式化できる．

$$\text{(P)} \quad \begin{array}{l} \text{minimize } \boldsymbol{f}(\boldsymbol{x}) \\ \text{subject to } \boldsymbol{x} \in X = \{\boldsymbol{x} \in R^n | \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{o}\} \end{array}$$

ここで，$\boldsymbol{x} = (x_1, x_2, ..., x_n)$ は n 次元決定変数ベクトル，$\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$ は m 次元ベクトル値関数，$\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \cdots, g_k(\boldsymbol{x}))$ は k 次元ベクトル値制約関数である．

このように，多目的最適化問題は k 個の不等式制約条件の下で m 個の目的関数を同時に最小化する n 次元の決定変数ベクトルを求める問題として定式化される．なお，制約のない多目的最適化問題では $k = 0$ となる．ところが多目的最適問題では，目的関数がベクトル値関数であるため，目的関数値の間に半順序関係しか成り立たない．そのため，一般に全ての目的関数を同時に最小化する完全最適解は存在しない．そこで，多目的最適化問題の最適解として種々の解概念が提案されているが，その最も代表的な解がPareto最適解である．

Pareto最適解 $\boldsymbol{x}^*(\in X)$ は，以下の条件を満足する解である．

$$\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{x}^*) \text{ となる} \boldsymbol{x} \in X \text{ が存在しない．}$$

ここで，ベクトル値の大小関係として，以下の比較演算子を定義しておく．

$$\begin{aligned} \boldsymbol{f} \leqq \boldsymbol{f}^* &\Leftrightarrow \forall i f_i \leq f_i^* \\ \boldsymbol{f} \leq \boldsymbol{f}^* &\Leftrightarrow \forall i f_i \leq f_i^* \text{ and } \exists j f_j < f_j^* \\ \boldsymbol{f} < \boldsymbol{f}^* &\Leftrightarrow \forall i f_i < f_i^* \end{aligned}$$

3 Simplex 法

Simplex 法は，R^n 上に幾つかの点を幾何的に配置し，それらの点での目的関数の値を比較することにより，最適解を探索する方法で，制約なし単一目的非線形最適化問題に対する直接探索による最適化手法の一つである．目的関数はスカラー値となるので，ここでは単に $f(\boldsymbol{x})$ と書くことにする．

R^n 上の $(n+1)$ 個のアフィン独立な点の凸包(単体)を生成し，単体の頂点を $\boldsymbol{x}^i \in R^n (i = 1, 2, ..., n+1)$ とし，頂点全体の集合を $U = \{\boldsymbol{x}^i\}$ とする．f の最小値を与える最良の頂点 $\boldsymbol{x}^l$，f の最大値を与える最悪の頂点 $\boldsymbol{x}^h$，f の2番目に大きな値を与える頂点 $\boldsymbol{x}^s$ を以下のように定義する．

$$\begin{aligned} \boldsymbol{x}^l &= \arg\min_i f(\boldsymbol{x}^i) \\ \boldsymbol{x}^h &= \arg\max_i f(\boldsymbol{x}^i) \\ \boldsymbol{x}^s &= \arg\max_{i \neq h} f(\boldsymbol{x}^i) \end{aligned}$$

さらに，$\boldsymbol{x}^h$ 以外の頂点から生成される図心を以下のように定義する．

$$\boldsymbol{x}^0 = \frac{1}{n} \sum_{i \neq h} \boldsymbol{x}^i$$

以上の点をもとにして，以下のような基本的手続きを定義する．(図1 参照)

鏡映 (reflection) $\boldsymbol{x}^h$ の $\boldsymbol{x}^0$ 方向への鏡映点 $\boldsymbol{x}^r$ を生成

$\boldsymbol{x}^r = (1+\alpha)\boldsymbol{x}^0 - \alpha \boldsymbol{x}^h \qquad (\alpha > 0)$

拡張 (expansion) $\boldsymbol{x}^h$ の $\boldsymbol{x}^r$ 方向への拡張点 $\boldsymbol{x}^e$ を生成

$\boldsymbol{x}^e = \gamma \boldsymbol{x}^r + (1-\gamma)\boldsymbol{x}^0 \qquad (\gamma > 1)$

収縮 (contraction) $\boldsymbol{x}^h$ の $\boldsymbol{x}^0$ 方向への収縮点 $\boldsymbol{x}^c$ を生成

$\boldsymbol{x}^c = \beta \boldsymbol{x}^h + (1-\beta)\boldsymbol{x}^0 \qquad (0 < \beta < 1)$

縮小 (reduction) 全頂点を $\boldsymbol{x}^l$ 方向へ縮小

$\boldsymbol{x}^i = \frac{1}{2}(\boldsymbol{x}^i + \boldsymbol{x}^l)$

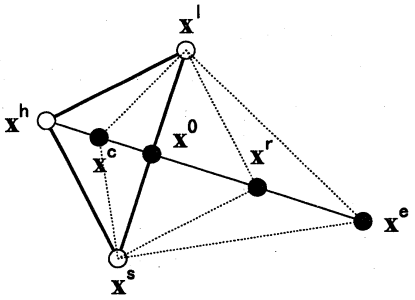

図1: Simplex 法の操作

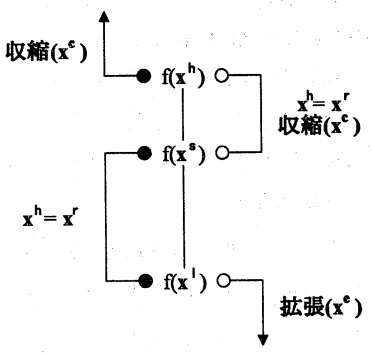

図2: Simplex 法の処理の概要

Simplex 法では，鏡映点での値 $f(\boldsymbol{x}^r)$ が図2 に示した区間のどこに位置するかによって処理を変える．すなわち，$f(\boldsymbol{x}^r)$ が最良点 $\boldsymbol{x}^l$ での値より良ければ拡張，2番目に悪い頂点 $\boldsymbol{x}^s$ での値と同じかそれより良ければ $\boldsymbol{x}^h$ を $\boldsymbol{x}^r$ で置換，最悪点 $\boldsymbol{x}^h$ での値より良ければ $\boldsymbol{x}^h$ を $\boldsymbol{x}^r$ で置換した後収縮，$\boldsymbol{x}^h$ での値と同じかそれより悪いならば収縮を行う．

このアルゴリズムをC言語的に表現したものを以下に記述する．なお，引数 U は頂点の全体集合である．

```
simplex_scalar(U)
{
  while(収束条件を満足しない) {
    x^l=U 中で最良の頂点;
    x^h=U 中で最悪の頂点;
    x^s=U 中で2番目に悪い頂点;
    x^0=U-{x^h} の図心;
    x^r=(1+α)x^0-αx^h;
    if(f(x^r) < f(x^l)) {
      x^e=γx^r+(1-γ)x^0;
      if(f(x^e) < f(x^l)) x^h=x^e;
      else x^h=x^r;
    }
    else if(f(x^r) ≤ f(x^s)) x^h=x^r;
    else {
      if(f(x^r) < f(x^h)) x^h=x^r;
      x^c=βx^h+(1-β)x^0;
      if(f(x^c) < f(x^h)) x^h=x^c;
      else
        for(all x^i in U) x^i=1/2(x^i+x^l);
    }
  }
}
```

通常，頂点の全体集合 U の初期値は，頂点間の距離がすべて等しい基準形の単体を用いる．収束条件は，以下のように各頂点における目的関数の値の差が十分に小さくなったときに満たされる．

$$\sqrt{\frac{1}{n+1}\sum_i (f(\boldsymbol{x}^i)-f^0)^2} \le \varepsilon \qquad \text{ここで，} f^0=\frac{1}{n+1}\sum_i f(\boldsymbol{x}^i)$$

4 Vector Simplex 法

前述のように，Simplex 法は基本的に全順序集合に対して適用できる手法である．本研究では，ベクトルの集合のような半順序集合に適用できるように Simplex 法を拡張した Vector Simplex 法を提案する．Vector Simplex 法は，制約のない多目的非線形最適化問題を解く際に利用することができる．

Vector Simplex 法では，頂点の全体集合 U は1つの解に収束するのではなく，Pareto 最適解集合へ収束する．このため，全体集合 U は $n+1$ 個の頂点の集合ではなく，$n+1$ 個以上の点を生成し，鏡映，拡張，収縮，縮小などの操作は全体集合 U から選択した $n+1$ 個の頂点に対して行う．

半順序集合では一般に最良解や最悪解を一意的に決定することはできないため，最良解に対応する「より良い頂点のない頂点の集合 U^l」，最悪解に対応する「より悪い頂点のない頂点の集合 U^h」，2番目に悪い解に対応する「それ以外の頂点の集合 U^s」を以下のように定め，U^h の要素を順次改善して行くことによって Pareto 最適解集合を求める．

$$\begin{aligned} U^l &= \{\boldsymbol{x}^l \in U \,|\, \boldsymbol{f}(\boldsymbol{x}) \le \boldsymbol{f}(\boldsymbol{x}^l) \text{ となる} \boldsymbol{x} \in U \text{が存在しない}\} \\ U^h &= \{\boldsymbol{x}^h \in U \,|\, \boldsymbol{f}(\boldsymbol{x}^h) \le \boldsymbol{f}(\boldsymbol{x}) \text{ となる} \boldsymbol{x} \in U - U^l \text{が存在しない}\} \\ U^s &= U - U^l - U^h \end{aligned}$$

Vector Simplex 法における処理の区分を図3 に示し，アルゴリズムについて以下に説明する．

1. 分割数と発生数の決定
   全体集合 U に属する頂点の第1成分 x_1 の存在領域(区間) D_1 を求める．区間 D_1 を等分割する分割数 d, 各部分区間に発生させる頂点数 m を決める．区間 D_1 を d 等分した部分区間 D_1^j, $j=1,2,\cdots,d$ の集合 $\mathfrak{D}$ を生成する．あらかじめ設定されたステップ数分，$(d,\ m)$ を決定し2. を実行し，その結果得られた U を Pareto 最適解集合とする．

2. 全体集合の再構成
   $\mathfrak{D}$ に属する部分区間 D_1^j を選択する．第1成分 x_1 が D_1^j に属する U 内の全点 $\boldsymbol{x}$ の第 i 成分の存在領域 D_i^j, $i=2,\cdots,n$ を求める．以後，$D^j = D_1^j \times D_2^j \times \cdots \times D_n^j$ と表す．D^j 内で無作為に生成した m 個の頂点を U に追加し，3.〜 10. の操作を実行する．$\mathfrak{D}$ 内の全ての部分区間が選択されたならば，1. に戻る．

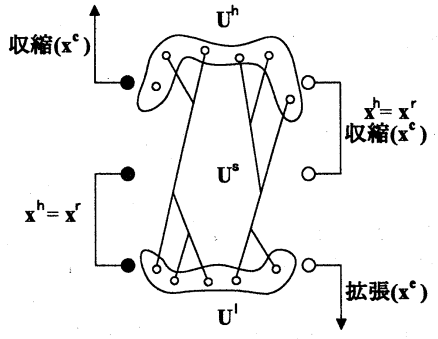

図 3: Vector Simplex 法の処理の概要

3. 部分集合の生成
   全体集合 U の部分集合 U^l, U^h, U^s を求める．$U^h = \emptyset$ ならば 2. に戻る．

4. 初期探索集合の構成
   U^h から1つ選択した頂点を $\boldsymbol{x}^h$ とし，$(U - \{\boldsymbol{x}^h\}) \cap D^j$ から選択した高々 n 個の頂点の集合を U^0 とする．両者を合わせた $n+1$ 個の頂点で探索の初期集合を構成する．

5. 鏡映
   鏡映点 $\boldsymbol{x}^r$ を求める．$\boldsymbol{f}(\boldsymbol{x}^r)$ との比較により以下のような処理を行う．

6. 拡張
   U^l 内に $\boldsymbol{x}^r$ より悪い頂点が存在する ($\boldsymbol{f}(\boldsymbol{x}^r) \leq \boldsymbol{f}(\boldsymbol{x})$ となる $\boldsymbol{x} \in U^l$ が存在する) ならば，拡張を行い $\boldsymbol{x}^e$ を求める．U^l 内に $\boldsymbol{x}^e$ より悪い頂点が存在すれば，$\boldsymbol{x}^h$ を $\boldsymbol{x}^e$ で置換し，そうでなければ $\boldsymbol{x}^h$ を $\boldsymbol{x}^r$ で置換する．これにより，$\boldsymbol{x}^h$ は次の段階では U^l の要素となり，収束に近づくことになる．

7. 置換
   $\boldsymbol{x}^r$ が U^l 内の頂点と同等 ($\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{x}^r)$ となる $\boldsymbol{x} \in U^l$ が存在しない)，または，U^s 内の頂点と同等かより良い ($\boldsymbol{f}(\boldsymbol{x}^r) \leqq \boldsymbol{f}(\boldsymbol{x})$ となる $\boldsymbol{x} \in U^s$ が存在する) ならば，$\boldsymbol{x}^h$ を $\boldsymbol{x}^r$ で置換する．これにより，$\boldsymbol{x}^h$ は U^l あるいは U^s の要素となる．

8. 収縮
   U^h 内に $\boldsymbol{x}^r$ より悪い頂点が存在すれば，$\boldsymbol{x}^h$ を $\boldsymbol{x}^r$ で置換する．この場合には，$\boldsymbol{x}^h$ は U^s の要素に相当する (*).
   収縮点 $\boldsymbol{x}^c$ を求める．$\boldsymbol{x}^c$ が $\boldsymbol{x}^h$ より改良されている（すなわち，$\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{x}^c)$ となる $\boldsymbol{x} \in U^l$ が存在しないか，または，$\boldsymbol{f}(\boldsymbol{x}^c) \leq \boldsymbol{f}(\boldsymbol{x})$ となる $\boldsymbol{x} \in U^h$ が存在する）ならば，$\boldsymbol{x}^h$ を $\boldsymbol{x}^c$ で置換する．これにより，$\boldsymbol{x}^h$ は U^l あるいは U^s の要素となる．

9. 縮小
   $\boldsymbol{x}^c$ が $\boldsymbol{x}^h$ より改良されていなければ，縮小を行う．ここでは1つの解へ収束するのではないことを考慮して，$\boldsymbol{x}^h$ のみを U^l の要素に近づける．U^h の要素である $\boldsymbol{x}^h$ には，明らかに U^l 中に $\boldsymbol{f}(\boldsymbol{x}^l) \leq \boldsymbol{f}(\boldsymbol{x}^h)$ となる $\boldsymbol{x}^l$ が存在する．また (*) の場合にも，$\boldsymbol{x}^h$ は U^s の要素に相当するので，条件を満足する $\boldsymbol{x}^l$ が存在する．このような $\boldsymbol{x}^l$ は1つ以上存在するので，その1つを選択して縮小を行う．したがって，U^h に属していた頂点は，U^l あるいは U^s の要素となるか，U^l の要素に近づくことになる．

10. 3. へ戻る

このように，Vector Simplex 法のアルゴリズムは，Simplex 法における頂点 $\boldsymbol{x}^l$, $\boldsymbol{x}^s$, $\boldsymbol{x}^h$ を集合 U^l, U^s, U^h に対応させたものとなっている．このアルゴリズムをC言語的に表現したものを以下に示す．

```
vector_simplex(U)
{
   for(s=1; s ≤ S_max; s++) {
       (d,m)=ξ(s);
       区間D_1=Uに属する全頂点のx_1の存在範囲;
       D_1^j=区間D_1をd等分した部分区間 (j = 1, 2,···,d);
       for(j=1; j ≤ d; j++) {
           D_i^j=Uに属する全頂点xの第i成分x_iの存在範囲(ただし, x_1 ∈ D_1^j, i = 2,···, n);
           D^j = D_1^j × D_2^j × ··· × D_n^j;
           U=U ∪ {D^j 内で無作為に発生させたm個の頂点};
           vector_simplex_area(U, D^j);
} } }
vector_simplex_area(U, D)
{
   while(1) {
       U^l=U内でより良い点が存在しない点の集合;
       U^h=U − U^l内でより悪い点が存在しない点の集合;
       if(U^h==∅) break;
       U^s=U-U^l-U^h;
       x^h ∈ U^hを選択;
       U^0 = (U − {x^h}) ∩ Dから選択したn個の頂点;
       x^0=U^0の図心;
       x^r=(1 + α)x^0 − αx^h;
       if(f(x) ≥ f(x^r)となるx ∈ U^lが存在する){
          x^e=γx^r + (1 − γ)x^0;
          if(f(x) ≥ f(x^e)となるx ∈ U^lが存在する)
             x^h=x^e;
          else
             x^h=x^r;
       }
       else if(f(x) ≤ f(x^r)となるx ∈ U^lが存在しない or
                f(x)≥f(x^r)となるx ∈ U^sが存在する)
          x^h=x^r;
       else {
          if(f(x) ≥ f(x^r)となるx ∈ U^hが存在する) x^h=x^r;
          x^c=βx^h + (1 − β)x^0;
          if(f(x) ≤ f(x^c)となるx ∈ U^lが存在しない or
             f(x) ≥ f(x^c)となるx ∈ U^hが存在する)
             x^h=x^c;
          else {
             x^l ∈ {x ∈ U^l|f(x) ≤ f(x^h)}を選択;
             x^h=(x^h + x^l)/2;
          } } } }
```

ただし，vector_simplexの引数であるUは初期集合，mは区間内で新たに追加する頂点数，S_{max}は最大ステップ数，$\xi(s)$はステップ数sから分割数dと追加頂点数mの組を決定する関数である．

5 制約のない多目的最適化問題

Vector Simplex 法の有効性を調べるために，以下のような制約のない多目的最適化問題に適用したときの結果について検討する．以降，Vector Simplex 法のパラメータについては，$S_{\max}=3$, $\xi(1)=(1,0)$, $\xi(2)=(10,10)$, $\xi(3)=(20,10)$, $\alpha=1$, $\beta=0.5$, $\gamma=2$ とする．

例1 minimize f_1, f_2

$$f_1(x)=x_1^2+x_2^2,\ f_2(x)=(x_1-1)^2+(x_2-1)^2$$

全体集合Uの初期値を原点を通る半径4の円周上に均等に配置された50点に設定し，このときの第1段階の結果を図4, 5 に示す．

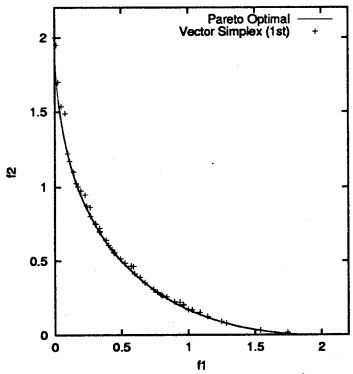

図4: 第1段階のf_1-f_2グラフ

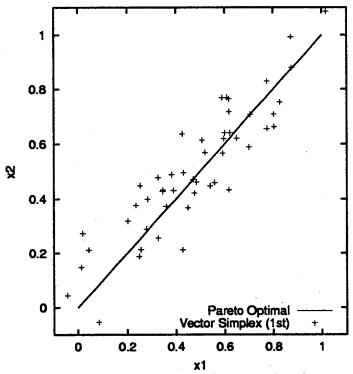

図5: 第1段階のx_1-x_2グラフ

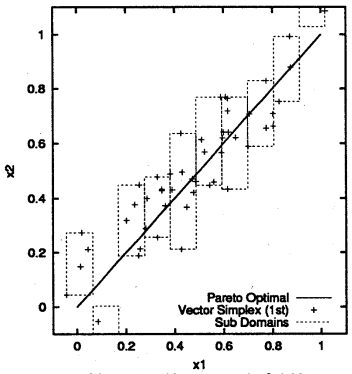

図6: 第2段階への分割状況

左側が目的関数(f_1,f_2)，中央が決定変数(x_1,x_2)のグラフであり，Pareto最適解集合を線分で，Vector Simplex 法の第1段階により求めたUを点で表している（以下同様）．目的関数のグラフからは，Pareto最適解集合の近似値が得られているように見えるが，例1におけるPareto最適解集合は，$\{(x_1,x_2)|x_1=x_2, 0\le x_1,x_2\le 1\}$ であり，決定変数のグラフからみると十分な精度が得られていると判断することは難しい．

第2段階では，Uに属する点を10分割し，各領域に無作為に10点ずつ生成しUに追加する．分割の様子を図6に示し，第2段階での，結果を図7，8に示す．

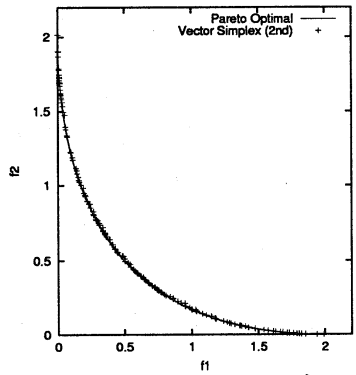

図7: 第2段階のf_1-f_2グラフ

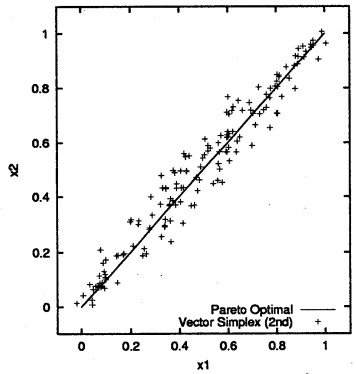

図8: 第2段階のx_1-x_2グラフ

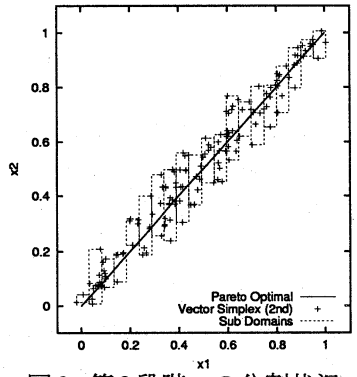

図9: 第3段階への分割状況

Uが第1段階に比して，かなりPareto最適解集合を近似している様子が分かる．第3段階では，Uを20分割し，各領域に10点を生成しUに追加する．分割の様子を図9に示し，得られた結果を図10，11に示す．図10，11から，目的関数だけでなくPareto最適解についてもかなりよい近似が得られた様子が分かる．

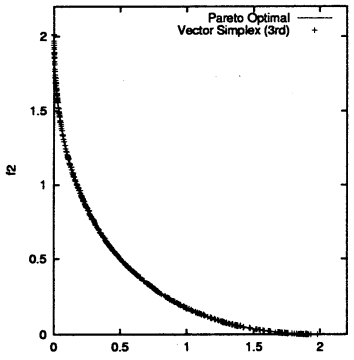

図10: 第3段階の$f_1 - f_2$グラフ

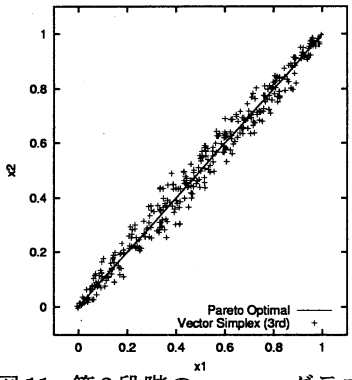

図11: 第3段階の$x_1 - x_2$グラフ

6 考察

本章では，例1に対してスカラー化法とSimplex法を組み合わせてPareto最適解集合を求めた結果と，Vector Simplex法による前章の結果を比較する．

ここではスカラー化法として加重平均法を採用する．したがって，ベクトル値関数である目的関数$\boldsymbol{f}(\boldsymbol{x})$は与えられた重みベクトル$\boldsymbol{w} = (w_1, \cdots, w_m)$によって以下の実数値関数$f(\boldsymbol{x})$に変換される．

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} w_i f_i(\boldsymbol{x}) \qquad \text{ただし} w_i \geq 0,\ i = 1, \cdots, m,\ \sum_{i=1}^{m} w_i = 1 \text{である．}$$

両手法による実験結果を表1に示す．

方法	解候補数	総評価回数	評価回数/解	平均誤差	最大誤差
Vector Simplex(1st)	50	388	7.76	0.0941	0.2548
Vector Simplex(2nd)	150	708	4.72	0.0493	0.1667
Vector Simplex(3rd)	350	1,256	3.59	0.0325	0.1108
Simplex	50	2,146	42.92	0.0027	0.0075

表 1: Vector Simplex と Simplex法の比較

ここで，1st, 2nd, 3rdはVector Simplex法の各段階数，Simplexは加重平均法＋Simplex法（以後Simplex法と表す)を意味し，解候補数は求められた解候補の数，総評価回数は目的関数の評価回数，評価回数/解は1解候補当たりの目的関数の評価回数，平均誤差は$|x_1 - x_2|$の平均値，最大誤差は$|x_1 - x_2|$の最大値を表す．

表1から，Vector Simplex法はSimplex法に比べて精度の点では及ばないが，1個の解候補を求めるために要する目的関数の評価回数は非常に少ないことが分かる．このことから，Vector Simplex法では目的関数値の評価に大きな計算コストを必要とする場合や目的関数の個数mが大きくなり最低限必要となるPareto最適解の個数が多くなるような場合に有効であると考えられる．

次に，例1の目的関数$f_2(\boldsymbol{x})$を10倍した関数に置き換えた次のような問題についてVector Simplex法とSimplex法による結果を比較する．ただし，その他の実行条件は例1の場合と全く同一である．

例2 minimize f_1, f_2

$$f_1(\boldsymbol{x}) = x_1^2 + x_2^2,\ f_2(\boldsymbol{x}) = 10\{(x_1 - 1)^2 + (x_2 - 1)^2\}$$

この問題に対するVector Simplex法による結果を図12，13に，Simplex法による結果を図14，15にそれぞれ示す．

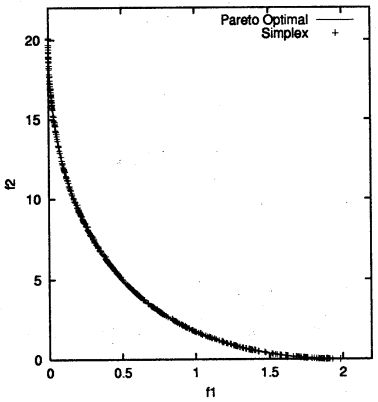

図12: Vector Simplex法 $f_1 - f_2$ グラフ

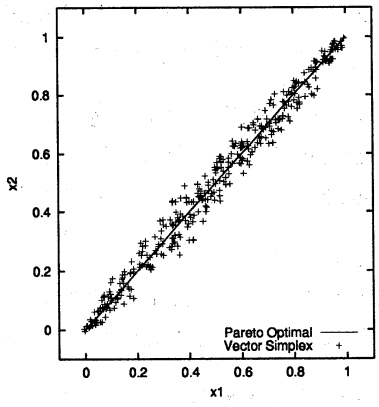

図13: Vector Simplex法の $x_1 - x_2$ グラフ

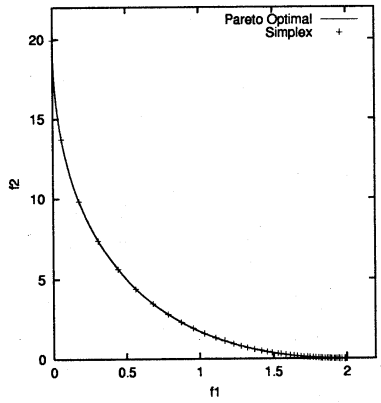

図14: Simplex法の $f_1 - f_2$ グラフ

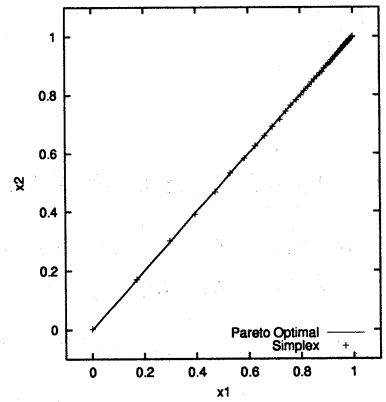

図15: Simplex法の $x_1 - x_2$ グラフ

図12－15から分かるように，Vector Simplex法の結果は例1の場合と全く変わらない．一方，Simplex法は，結果にかなり偏りが見られる．これは，加重平均によって得られた単一目的関数において $f_2(\boldsymbol{x})$ の値が占めるウエイトが大きくなったためと考えられる．すなわち，各目的関数値の尺度に開きがあり値の大きな目的関数に依存した偏った探索が行われたと考えられる．そのため，各目的関数値の尺度を調整するための変換が必要である．これは，各目的関数値の評価に必要な計算コストが大きい場合や目的関数の個数が大きい場合に，変換のための計算コストを増大させる．

一方，Vector Simplex法ではベクトル値関数における半順序関係によって目的関数を評価するため，各目的関数値の尺度に影響を全く受けず，目的関数の単位や尺度変換の必要が全くない．

7 おわりに

非線形最適化手法である Simplex 法を半順序集合に適応できるように拡張し，制約のない多目的非線形最適化問題においてPareto最適解集合を直接求めることが可能な Vector Simplex 法を提案した．Vector Simplex 法では，先ず少ない点によって，ラフな解集合を求め，その集合に属する頂点の近傍に新たな頂点を生成し，再度，解集合を求めるという処理を繰り返すことにより，かなり精度の高いPareto最適解集合を求めることができることを示した．

また，最もよく採用されるスカラー化法である加重平均法による結果と比較することにより，Vector Simplex法によって高速にかなり精度の高いPareto最適解集合を得られることを示した．さらに，ベクトル間の半順序関係による評価を行っているため目的関数間の尺度に独立に均一なPareto最適解集合を得られることを示した．

Vector Simplex 法は Simplex 法と同様な実践的手法である．そこで，今後は以下のような方針でVector Simplex 法の効果について研究を進めてゆく予定である．

1. 制約付き多目的非線形最適化問題への対応
   Vector Simplex法は制約なし多目的非線形最適化問題に対する最適化手法である．そのため，制約付き多目的非線形最適化問題にも適用可能なようにVector Simplex法を改良する必要がある．

2. 他の手法との比較
   最初に触れたように，Pareto最適解集合を直接求める方法として遺伝的アルゴリズムを利用した方法がある．これらの方法との比較がまだ不十分なため，より詳細な比較を行う必要がある．

3. 制御パラメータの最適化
   例えば倒立振子における角度制御と位置制御のように，制御目標が複数あり，どちらを優先すべきかを判断しにくいような制御系に対して，制御システムのパラメータを最適化するためにVector Simplex法を利用する．

参考文献

[1] 前田隆: 多目的意志決定と経済分析, 牧野書店 (1996).

[2] 阪井節子, 高濱徹行: 非線形最適化手法を用いた倒立振子ファジィ制御規則の学習, 第13回ファジィシステムシンポジウム講演論文集, pp. 61–64 (1997).

[3] 西川緯一, 三宮信夫, 茨木俊秀: 岩波講座 情報科学 最適化, 岩波書店 (1982).

[4] 坂和正敏: 非線形システムの最適化, 森北出版 (1986).

[5] 北野宏明編: 遺伝的アルゴリズム 2 , 産業図書 (1995).

[6] Schaffer, J.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proc. of the 1st ICGA*, pp. 93–100 (1985).

[7] 村田忠彦, 石渕久生, 田中英夫: 遺伝的アルゴリズムによるフローショップ・スケジューリングと多目的最適化問題への応用, 計測自動制御学会論文集, Vol. 31, No. 5, pp. 583–590 (1995).

[8] Goldberg, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).

[9] Fonseca, C. and Fleming, P.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, *Proc. of the 5th ICGA*, pp. 416–423 (1993).

[10] Louis, S. and Rawlins, G.: Pareto Optimality, GA-Easiness and Deception, *Proc. of the 5th ICGA*, pp. 118–123 (1993).

[11] Kowalik, J. and Osborne, M.: *Methods for Unconstrained Optimization Problems*, American Elsevier Publishing Company (1968).

[12] 今野浩, 山下浩: 非線形計画法, 日科技連出版社 (1978).